User's Guide: FMS Atmospheric Dynamical Cores

Lori Thompson <Lori.Thompson@noaa.gov>

Table of Contents

Quickstart guide: FMS Atmospheric Dynamical Cores
[quickstart.html]

1. Introduction

1.1. What is an atmospheric dynamical core?

We divide a global atmospheric model into a "dynamical core" and a set of "physics" modules, the combination of which is sufficient to integrate the state of the atmosphere forward in time for a given time interval. The dynamical core must be able to integrate the basic fluid equations for an inviscid, adiabatic ideal gas over an irregular rotating surface forward in time. Included in the dynamical core is the horizontal and vertical advection, horizontal subgrid scale mixing, and the time differencing.

Our operational definition of a global atmospheric dynamical core is a module, or set of modules, which is capable of integrating a particular benchmark calculation defined by Held and Suarez (1994)[1] so as to obtain a statistically steady "climate". Model physics is replaced by very simple linear relaxation of temperature to a specified spatial structure and the winds near the surface are relaxed to zero. Because the resulting flow is turbulent and cascades variance to small horizontal scales, either explicit or implicit horizontal mixing is required to obtain a solution, and, as stated above, is considered to be part of the dynamical core.

1.2. The available cores

1.2.1. Finite difference core (B-grid core)

The global, hydrostatic finite difference dynamical core, also called the B-grid core, was developed from models described in Mesinger, et al. (1988) [2] and Wyman (1996) [3]. The horizontal grid is the Arakawa B-grid and a hybrid sigma/pressure vertical coordinate is used. The step-mountain (eta coordinate) option is no longer supported.

1.2.2. Spectral core

The spectral dynamical core is a "plain vanilla" version of the classic Eulerian spectral model, with a spherical harmonic basis, for integrating the primitive equations on the sphere. The option of advecting tracers with a finite-volume grid point scheme is also available. Barotropic (2D non-divergent) and shallow water spherical models are also provided.

1.2.3. Finite-volume core

The finite-volume (FV) core is described in Lin 2004.[4] The horizontal grid is currently based on the regular latitude-longitude grid. The vertical coordinate internal to the FV core is fully Lagrangian with remapping to a Euler coordinate as used in the physical parameterizations.

1.3. Support, feedback, user contributions

We will try our best to respond to your support requests and bug reports quickly, given the limits of our human resources devoted to this project. Please use the mailing lists (`<oar.gfdl.fms@gfdl.noaa.gov>` and `<oar.gfdl.fms-atmos@gfdl.noaa.gov>`) as the forum for support requests: browse the mailing lists for answers to your questions, and post new questions directly to the mailing list. We would also appreciate it if you could answer other users' questions, especially those related to site and platform configuration issues that you may have encountered. We will provide very limited support for platforms not listed in Section 1.4, “Portability”,

1 Held, I. M., and M. J. Suarez, 1994: **A proposal for the intercomparison of the dynamical cores of atmospheric general circulation models.** *Bull. of the Am. Meteor. Soc.*, 75(10), 1825--1830. Download PDF [http://www.gfdl.noaa.gov/~gth/netscape/1994/ih9401.pdf]

2 Mesinger, F., Z. I. Janjic, S. Nickovic, D. Gavrilov and D. G. Deaven, 1988: **The step-mountain coordinate: Model description and performance for cases of Alpine lee cyclogenesis and for a case of an Appalachian redevelopment.** *Mon. Wea. Rev.*, 116, 1493--1518.

3 Wyman, B. L., 1996: **A step-mountain coordinate general circulation model: Description and validation of medium-range forecasts.** *Mon. Wea. Rev.*, 124, 102--121.

4 Lin, S-J., 2004: **A "vertically Lagrangian" finite-volume dynamical core for global models.** *Mon. Wea. Rev.*, 132(10), 2293--2307. Download PDF [http://www.gfdl.noaa.gov/reference/bibliography/2004/sjl0402.pdf]

and for modifications that you make to the released code.

1.4. Portability

Our commitment at any given time is only on those platforms where we have adequate access for our own thorough testing. We will add supported platforms as we can.

The platforms we support at present are the following:

```
1) SGI
   Chipset: MIPS
   OS: Irix 6.5
   Compiler: MIPSPro version 7.3.1.2 or higher
   Libraries: Message Passing Toolkit (MPT) version 1.5.1.0 or higher.
              netCDF version 3.4 or higher (64-bit version)

2) Linux
   Chipset: AMD
   OS: GNU/Linux
   Compiler: Intel Fortran Compiler Version 9.0-027
   Libraries: MPI-1 (e.g MPICH-1.2.5)
              netCDF version 3.4 or higher (64-bit version)
```

1.5. FMS Licensing

The Flexible Modeling System (FMS [http://www.gfdl.noaa.gov/~fms]) is free software; you can redistribute it and/or modify it and are expected to follow the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

FMS is distributed in the hope that it will be useful, but *WITHOUT ANY WARRANTY*; without even the implied warranty of *MERCHANTABILITY* or *FITNESS FOR A PARTICULAR PURPOSE*. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this release; if not, write to:

Free Software Foundation, Inc.
59 Temple Place, Suite 330
Boston, MA 02111-1307
USA
or see: http://www.gnu.org/licenses/gpl.html

2. Details of the code

2.1. Where to find documentation?

In addition to this web page, additional documentation for the atmospheric dynamical cores may be found in these documents:

- Technical descriptions of the available cores (PDF files)

  - Finite differencing used by the B-grid dynamical core [../src/atmos_bgrid/documentation/bgrid.pdf]
  - Spectral dynamical core [../src/atmos_spectral/documentation/spectral_core.pdf]
  - Spectral barotropic version [../src/atmos_spectral_barotropic/barotropic.pdf]
  - Spectral shallow water version [../src/atmos_spectral_shallow/shallow.pdf]
- HTML documentation files
  - Quickstart guide [quickstart.html]
  - Index of atmospheric dynamical core documentation [index.html]
  - B-grid dynamical core supplementary documentation [../ src/ atmos_bgrid/ documentation/ bgrid_supdoc.html]

2.2. Overview of the finite difference core

Key features of the finite difference dynamical core are:

- hydrostatic
- latitude-longitude grid with Arakawa B-grid staggering
- hybrid sigma-pressure vertical grid
- prognostic variables are the zonal and meridional wind components, surface pressure, temperature, and an arbitrary number of tracers
- two-level time differencing scheme
  - gravity waves are integrated using the forward-backward scheme
  - split time differencing is used for longer advective and physics time steps
- pressure gradient options:
  - Simmons and Burridge (1981) [5] energy, angular momentum conserving scheme (default)
  - Lin (1997) [6] finite-volume method
- default advection option uses centered spatial differencing
  - modified Euler backward time differencing for stability
  - second and fourth-order options
- vertical advection option for piecewise linear (van Leer) or parabolic (PPM) finite volume schemes
- grid point noise and the two-grid-interval computational mode of the B-grid are controlled with linear horizontal damping
- Fourier filtering in high latitudes of the shortest resolvable waves so that a longer time step can be taken (filter is applied to the mass divergence, horizontal omega-alpha term, horizontal advective tendencies, and the momentum components)
- optional sponge at top model level to reduce the reflection of waves
- optional divergence damping with options for second or fourth-order (much less dissipative)
- energy conservation guaranteed for long climate runs with a global energy correction to temperature

5Simmons, A. J. and D. M. Burridge, 1981: **An energy and angular-momentum conserving vertical finite-difference scheme and hybrid vertical coordinates.** *Mon. Wea. Rev.*, 109, 758–766.

6Lin, S.-J., 1997: **A finite-volume integration method for computing pressure gradient force in general vertical coordinates.** *Quart. J. Roy. Meteor. Soc.*, 123, 1749–1762.

General features are:

- written using Fortran 90
- meets the code standards of the FMS [http://www.gfdl.noaa.gov/~fms]
- two-dimensional domain decomposition on the longitude/latitude axes
- array storage by longitude (fastest varying), latitude, level, and tracer number; using indexing (i,j,k,n)

2.3. Overview of the spectral core

Key features of the spectral dynamical core are:

- standard spherical harmonic dynamical core for integrating hydrostatic equations for an ideal gas over the sphere. Prognostic variables are vorticity and divergence of horizontal flow, temperature, logarithm of surface pressure, and an arbitrary number of tracers (See Durran, Numerical Methods for Wave Equations in Geophysical Fluid Dynamics, Springer Verlag, 1999)
- shallow water and non-divergent barotropic models are also provided with pedagogical examples to help introduce users to this spectral framework
- standard semi-implicit time differencing for gravity waves
- vertical coordinate surfaces are defined, as in B-grid core, by the set of coefficients (A_k, B_k) where the pressure on the interfaces between layers is **p_k = A_k + B_k p_s**. By choosing these coefficients appropriately, one can transition from pure "sigma" coordinate (**A = 0**) near the surface to pressure coordinates aloft
- vertical differencing follows Simmons and Burridge (1981)
- vertical advection module for tracers shared with B-grid core
- horizontal tracer advection can be performed with standard spectral advection algorithm or with a finite volume scheme. The latter is currently limited to piecewise linear Van Leer, as implemented on the sphere by Lin and Rood (1996).[7] A piecewise parabolic scheme will be available shortly, sharing code with the grid core. Mass of tracer is not conserved exactly with either spectral or finite volume scheme in this context, but performance [#performance] in full physics atmospheric model is encouraging
- damping is Laplacian of vorticity, divergence, temperature, tracer raised to the n'th power; damping not needed for tracers when using finite volume advection
- sector option (m-fold symmetry in longitude) also available for dynamical studies

2.4. Overview of the finite-volume core

Key features of the finite-volume dynamical core are:

- Hydrostatic
- Latitude-longitude grid with a staggered two-grid system (C and D; see Lin and Rood 1997 [8]).
- Lagrangian control-volume vertical coordinate with a mass, momentum, and total energy conserving remapping to a Eulerian coordinate (e.g, the hybrid sigma-pressure used in the AM2 Model).
- Multiple levels of time splitting within the FV core, with the remapping time step the same as the physics time step. The tracer time step is normally the same as the remapping except when the meridional CFL condition is violated.

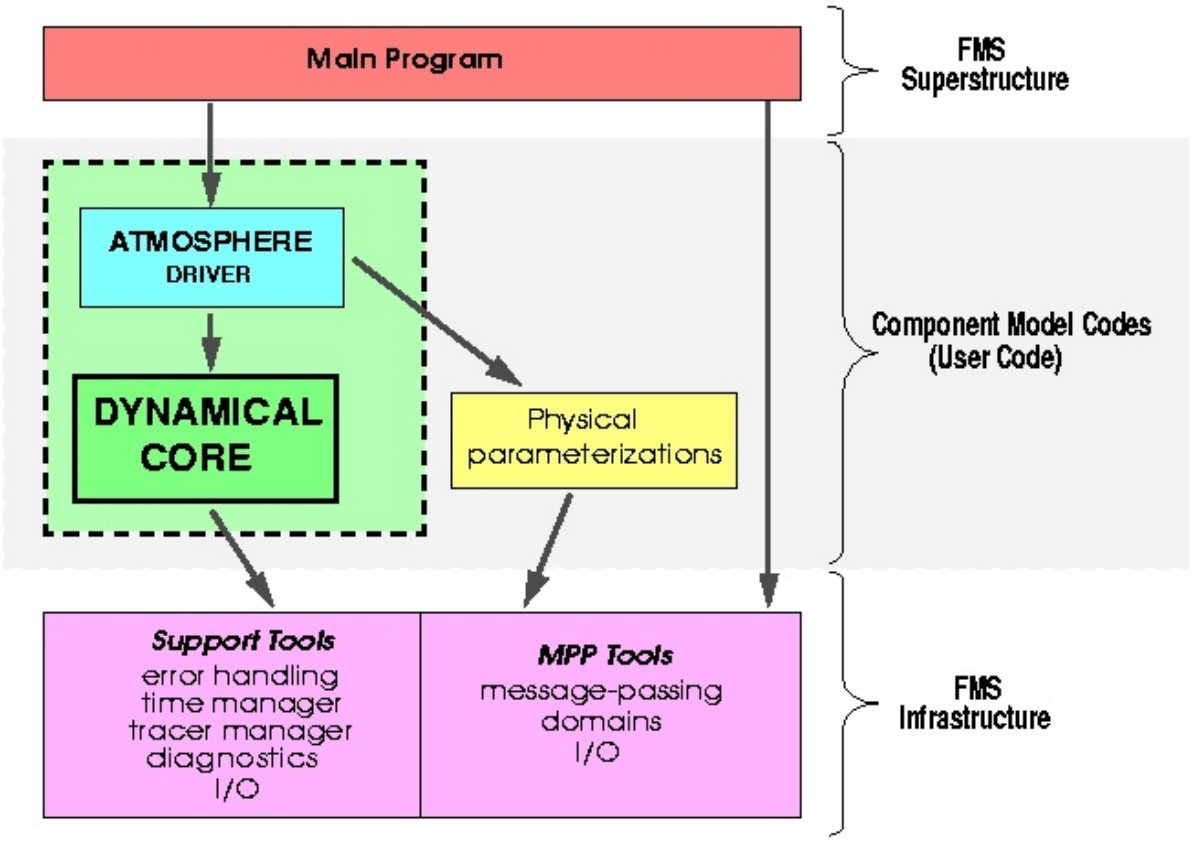
[7] Lin, S.-J. and R.B. Rood, 1996: **Multidimensional flux-form semi-Lagrangian transport schemes.** *Quart. J. Roy. Meteor. Soc.*, 123, 1749--1762.

[8] Lin, S.-J. and R.B. Rood, 1997: **An Explicit Flux-Form Semi-Lagrangian Shallow-Water Model on the Sphere,** *Quart. J. Roy. Meteor. Soc.*, 123(544), 2477--2498.

2.5. The dynamical core interface

We do not specify the precise interface expected of a dynamical core. Within FMS [http://www.gfdl.noaa.gov/~fms] we do specify the precise interface for the atmospheric model as a whole, so as to allow it to communicate with land, ocean, and ice models. Atmospheric models are generally constructed for each core individually from the dynamical core and individual physics modules. Sets of physics modules are bundled into physics packages that can be used to more easily compare models with the same physics and different cores (the Held-Suarez forcing is the simplest example of such a package) but we also recognize that different cores may require the physics to be called in distinct ways.

The B-grid and spectral dynamical cores share high-level superstructure code and low-level FMS infrastructure codes. The dynamical core is sandwiched between these levels. For the simple test cases provided with this release the superstructure only consists of a main program, but in more realistic models, drivers for component models and coupling software may also be considered part of the superstructure. The FMS [http://www.gfdl.noaa.gov/~fms] infrastructure codes, which include many commonly-used utilities, are used by both the dynamical core and the superstructure code. The following figure depicts this hierarchy of model codes.



The dynamical cores have the same user interface at the atmospheric driver level. Atmospheric drivers are constructed for each core for specific types of models. The drivers included with this public release are for running in a dynamical core only mode using simple forcing from a shared module (the Held-Suarez GCM benchmark model). Other drivers exist for running coupled models using full atmospheric physics in either AMIP mode or fully coupled to a realistic ocean, ice, and land model.

A user selects which dynamical core to use prior to compiling the model. A list of path names to the source code of a specific core and the FMS [http://www.gfdl.noaa.gov/~fms] shared codes is supplied to a compilation script. Refer to Section 5, “Preparing the runscript” for details on compiling the source code.

2.6. Shared components

1. FMS infrastructure

parallelization tools [../src/shared/mpp/mpp.html]	Simple routines that provide a uniform interface to different message-passing libraries, perform domain decomposition and updates, and parallel I/O on distributed systems.
I/O and restart file utilities [../src/shared/fms/fms.html]	Routines for performing commonly used functions and for reading and writing restart files in native or NetCDF [http://www.unidata.ucar.edu/packages/netcdf/] format.
time manager [../src/shared/time_manager/time_manager.html]	Simple interfaces for managing and manipulating time and dates.
time_interp [../src/shared/time_interp/time_interp.html]	Computes weight and dates indices for linearly interpolating between two dates.
diagnostics manager [../src/shared/diag_manager/diag_manager.html]	Simple calls for parallel NetCDF [http://www.unidata.ucar.edu/packages/netcdf/] diagnostics on distributed systems.
field manager/ [../src/shared/field_manager/field_manager.html]	Code to read entries from a field table [#fieldTable] and manage the simple addition of tracers to the FMS [http://www.gfdl.noaa.gov/~fms] code.
tracer manager [../src/shared/tracer_manager/tracer_manager.html]	
fast Fourier transform [../src/shared/fft/fft.html]	Performs simultaneous FFTs between real grid space and complex Fourier space.
topography [../src/shared/topography/topography.html]	Routines for creating land surface topography and land-water masks.
constants [../src/shared/constants/constants.html]	Sets values of physical constants and pi.
horiz_interp [../src/shared/horiz_interp/horiz_interp.html]	Performs spatial interpolation between grids.
axis_utils [../src/shared/axis_utils/axis_utils.html]	A set of utilities for manipulating axes and extracting axis attributes.
data_override [../src/shared/data_override/data_override.html]	Utility for spatial and temporal interpolation of data to the model grid and time.
memutils	Various operations for memory management.
platform [../src/shared/platform/platform.html]	Provides public entities whose value may depend on the operating system and compiler.

2. Atmospheric shared components

vertical advection [../src/atmos_shared/vert_advection/vert_advection.html]	Computes a tendency due to vertical advection for an arbitrary quantity.
forcing for Held-Suarez GCM benchmark [../src/atmos_param/hs_forcing/hs_forcing.html]	Routines for computing heating and dissipation for the Held-Suarez GCM benchmark integration of a dry GCM.
atmos_nudge	Routines for nudging of the atmospheric data

3. FMS superstructure

main program [../src/atmos_solo/atmos_model.html]	For running a stand-alone atmospheric model.

3. Acquiring source code

3.1. How to acquire source code

The Flexible Modeling System development team at GFDL uses a local implementation of GForge to serve FMS software, located at http://fms.gfdl.noaa.gov. In order to obtain the source code, you must register [https://fms.gfdl.noaa.gov/account/register.php] as an FMS user on our software server. After submitting the registration form on the software server, you should receive an automatically generated confirmation email within a few minutes. Clicking on the link in the email confirms the creation of your account.

After your account has been created, you should log in [https://fms.gfdl.noaa.gov/account/login.php] and request access to the FMS Atmospheric Dynamical Cores project. Once the FMS project administrator grants you access, you will receive a second e-mail notification. This email requires action on the part of the project administrator and thus may take longer to arrive. The email will contain instructions for obtaining the release package, which are described below.

The download will create a directory called `atm_dycores` in your current working directory containing the release package. The readme [../readme] file in the `atm_dycores` directory gives a brief overview of the package's directory structure and contents.

Sample output is also available for download. See Section 6.1, “Model output” for more information on the sample output.

3.2. What is GForge?

GForge [http://www.gforge.org] is an Open Source collaborative software development tool, which allows organization and management of any number of software development projects. It is designed for managing large teams of software engineers and/or engineers scattered among multiple locations. Gforge is available at http://www.gforge.org. General user documentation can be found at http://gforge.org/docman/?group_id=1.

4. Compiling the source code

4.1. The mkmf utility

The sample runscripts use the utility **mkmf** [../bin/mkmf.html], provided in the `atm_dycores/bin` directory, to create a makefile, and the **make** utility to compile the FMS source code. A listing of paths to all checked out source code files is included in the `path_names` files, located in each dynamical core directory under the `atm_dycores/exp/` [../exp] directory. The `path_names` file is used by the makefile utility **mkmf** to create a makefile. The `path_names.html` files included in the `atm_dycores/exp/$dycore` [../exp] directory are created for the user's convenience and contain links to all the relevant documentation files that have been checked out from the download along with the source code. A makefile is used to determine the source code dependencies and the order in which source code files will be compiled. **mkmf** ("make-make-file" or "make-m-f") is a tool written in perl5 that will construct a makefile from distributed source. The result of the **mkmf** utility is a single executable program. Note that **mkmf** is called automatically in the sample runscripts [#runscript].

mkmf [../bin/mkmf.html] has the ability to understand dependencies in f90, such as modules and use, the FORTRAN `include` statement and the cpp `#include` statement in any type of source. **mkmf** also places no restrictions on filenames, module names, etc. The utility supports the concept of overlays, where source is maintained in layers of directories with a defined precedence. In addition, **mkmf** can keep track of changes to cpp flags and knows when to recompile the affected source. This refers to files containing `#ifdefs` that have been changed since the last invocation.

The calling syntax is:

```
mkmf [-a abspath][-c cppdefs][-d][-f][-m makefile][-p program] [-t template][-v][-x][arg
```

```
    -a abspath     attaches the absolute path to the front of all relative paths to sou
    -c cppdefs     list of cpp #defines to be passed to the source files
    -d             debug flag
    -f             formatting flag
    -m makefile    name of makefile [#creatingMakefile] written
    -p program     final target name
    -t template    file containing a list of make macros or commands
    -v             verbosity flag
    -x             executes the makefile [#creatingMakefile] immediately
    args           list of directories and files to be searched for targets and depende
```

The debug flag is much more verbose than `-v` and used only if you are modifying **mkmf** itself. The formatting flag restricts the lines in the makefile [#creatingMakefile] to 256 characters. Lines exceeding 256 characters use continuation lines. If filenames are omitted for the options `[-m makefile]` and `[-p program]`, the defaults `Makefile` and `a.out` are applied. The list of make macro or commands contained in `[-t template]` are written to the beginning of the makefile.

4.2. Creating the makefile

When the **mkmf** [../bin/mkmf.html] utility is executed, it reads a template file and runs a list of make macros, commands and compilers. The template is a platform-specific file that contains standard compilation flags. Default template files are provided with the FMS [http://www.gfdl.noaa.gov/~fms] source code and are located in the `atm_dycores/bin` [../bin] directory. It is recommended that users set up their own compilation template specific for their platform and compiler. The template file contains the following variables:

```
    FC             compiler for FORTRAN files
    LD             executable for the loader step
    CPPFLAGS       cpp options that do not change between compilations
    FFLAGS         flags to the FORTRAN compiler
    LDFLAGS        flags to the loader step
    CFLAGS         flags to the C compiler
```

For example, the template file for the SGI Intel Fortran Compiler contains:

```
    FC = ifort
    LD = ifort
    CPPFLAGS = -I/usr/local/include
    FFLAGS = $(CPPFLAGS) -fno-alias -stack_temps -safe_cray_ptr -ftz -i_dynamic -assume
    LDFLAGS = -L/usr/local/lib -lnetcdf -lmpi -lsma
    CFLAGS = -D__IFC
```

An include file is any file with an include file suffix, such as `.H`, `.fh`, `.h`, `.inc`, which is recursively searched for embedded includes. **mkmf** [../bin/mkmf.html] first attempts to locate the include file in the same directory as the source file. If it is not found there, it looks in the directories listed as arguments, maintaining a left-to-right precedence. The argument list, `args`, is also treated sequentially from left to right. The default action for nonexistent files is to create null files of that name in the current working directory via the UNIX **touch** command. There should be a single main program source among the arguments to **mkmf** [../bin/mkmf.html], since all the object files are linked to a single executable.

The argument `cppdefs` should contain a comprehensive list of the cpp `#defines` to be preprocessed. This list is compared against the current "state", which is maintained in the file `.cppdefs` in the current working directory. If there are any changes to this state, **mkmf** will remove all object files affected by this change so that the subsequent make will recompile those files. The file `.cppdefs` is created if it does not exist. `.cppdefs` also sets the make macro `CPPDEFS`. If this was set in a template file and also in the `-c` flag to **mkmf**, the value in `-c`

takes precedence. Typically, you should set only `$CPPFLAGS` in the template file and `CPPDEFS` via **mkmf -c**.

To execute the **mkmf** [../bin/mkmf.html] utility, the user must locate the appropriate `path_names` file in the `atm_dycores/exp/$dycore` [../exp] directory. The script **list_paths**, in the `atm_dycores/bin` [../bin] directory, can be used to create a new `path_names` file containing a list of all source code files in a given directory. The user should set up the compilation template file and execute the **mkmf** utility. With the appropriate options illustrated in the sample runscripts, the user can call **mkmf** from the compilation directory and cause **mkmf** to call the **make** command automatically after the `Makefile` is generated.

4.3. Compiling without MPI

Underlying FMS [http:/ / www.gfdl.noaa.gov/ ~fms] is a modular parallel computing infrastructure. MPI (Message-Passing Interface) is a standard library developed for writing message-passing programs for distributed computing across loosely-coupled systems. Incorporated in the FMS [http:/ / www.gfdl.noaa.gov/ ~fms] source code is MPP [../src/shared/mpp/mpp.html] (Massively Parallel Processing), which provides a uniform message-passing API interface to the different message-passing libraries. Together, MPI and MPP establish a practical, portable, efficient, and flexible standard for message passing.

There are a number of freely available implementations of MPI that run on a variety of platforms. The MPICH implementation, developed by researchers at Argonne National Lab and Mississippi State University, runs on many different platforms, from networks of workstations to MPPs. If MPICH is installed, the user can compile the source code with MPI. If the user does not have MPICH or the communications library, the FMS [http:// www.gfdl.noaa.gov/ ~fms] source code can be compiled without MPI in one of two ways. If the makefile [#creatingMakefile] is created external to the runscript [#runscript], omit the `-c cppdefs` flag from the **mkmf** [../bin/mkmf.html] syntax. To compile the FMS [http://www.gfdl.noaa.gov/~fms] source code without MPI delete `-Duse_libMPI` from the `cppflags` variable.

When MPI is not used, the messages from `MPP_lib` may display but the data is being copied by MPP. Compiling the FMS [http://www.gfdl.noaa.gov/~fms] source code without MPI has been tested and the results show that 1 GB of memory is required for executing the code with and without MPI on a single PE.

5. Preparing the runscript

5.1. The runscript

Simple (csh shell) scripts are provided for running the atmospheric dynamical cores. These runscripts perform the minimum required steps to compile and run the models and are intended as a starting point for the development of more practical run scripts. The scripts for all available dynamical core models are located in each `atm_dycores/exp/$dycore` [../exp] directory.

Near the top of the scripts, variables are set to the full path name of the initial condition (optional), diagnostics table [#diagTable], tracer field table [#fieldTable] (optional), compilation directory, and the **mppnccombine** [#mppnccombine] script. The script proceeds to compile and link the source code, create a working directory, and copy the required input data into the working directory. The **mpirun** command is then used to run the model. The final step for multi-processor runs, is to combine domain-decomposed diagnostics files into global files.

The default the scripts are set to run one or two days on a single processor. The number of processors used is controlled by a variable near the top of the scripts. Users may want to increase the number of processors to decrease the wallclock time needed for a run. The run length (in days) and the atmospheric time step, `dt_atmos`, in seconds is controlled by namelist `&main_nml` which is set directly in the runscripts for convenience.

To compile and link the model codes a template file provides platform dependent parameters, such as the location of the netCDF library on your system, to a compilation utility called **mkmf** [../bin/mkmf.html]. Sample template files for various platforms are provided in the `bin` directory. More information on **mkmf** and compiling the

source code can be found in Section 4, “Compiling the source code”.

The sample scripts compile the model code with the MPI library and execute using the mpirun command. Refer to Section 4.3, “Compiling without MPI” for issues related to the MPI implementation and for compiling without MPI. The **mpirun** command is specific to Silicon Graphics machines, and users may need to change this to run on other platforms.

The following sections will describe some of the steps needed to understand and modify the simple runscripts.

5.2. The diagnostic table

The FMS [http://www.gfdl.noaa.gov/~fms] diagnostics manager [../src/shared/diag_manager/diag_manager.html] is used to save diagnostic fields to netCDF [http://www.unidata.ucar.edu/packages/netcdf/] files. Diagnostic output is controlled by specifying file and field names in an ASCII table called `diag_table`. The diagnostic tables supplied with the dynamical cores are found in each `atm_dycores/exp/$dycore` [../exp] directory in a file called `diag_table`. The bgrid, spectral, and FV test cases use a standard table for the Held-Suarez benchmark test case. In the runscript [#runscript], the user specifies the full path to the appropriate diagnostics table, and it is copied to the file `diag_table` in the directory where the model is run.

The diagnostic table consists of comma-separated ASCII values and may be edited by the user. The table is separated into three sections: the global section, file section and field section. The first two lines of the table comprise the global section and contain the experiment title and base date. The base date is the reference time used for the time axis. For the solo dynamical cores the base date is irrelevant and is typically set to all zeroes. The lines in the file section contain the file name, output frequency, output frequency units, file format (currently, only netCDF), time units and long name for the time axis. The last section, the field section, contains the module name, field name, output field name, file name, time sampling for averaging (currently, all time steps), time average (.true. or .false.), other operations which are not implemented presently and the packing value. The packing value defines the precision of the output: 1 for double, 2 for floating, 4 for packed 16-bit integers and 8 for packed 1-byte. Any line that begins with a "#" is a comment.

A sample diagnostic table is displayed below.

```
"Model results from the Held-Suarez benchmark"
0 0 0 0 0 0
#output files
"atmos_daily",    24, "hours", 1, "days", "time",
"atmos_average",  -1, "hours", 1, "days", "time",
#diagnostic field entries.
 "dynamics",    "ps",             "ps",             "atmos_daily",    "all", .false., "n
 "dynamics",    "bk",             "bk",             "atmos_average",  "all", .false., "n
 "dynamics",    "pk",             "pk",             "atmos_average",  "all", .false., "n
 "dynamics",    "zsurf",          "zsurf",          "atmos_average",  "all", .false., "n
 "dynamics",    "ps",             "ps",             "atmos_average",  "all", .true.,  "n
 "dynamics",    "ucomp",          "ucomp",          "atmos_average",  "all", .true.,  "n
 "dynamics",    "vcomp",          "vcomp",          "atmos_average",  "all", .true.,  "n
 "dynamics",    "temp",           "temp",           "atmos_average",  "all", .true.,  "n
 "dynamics",    "omega",          "omega",          "atmos_average",  "all", .true.,  "n
 "dynamics",    "div",            "div",            "atmos_average",  "all", .true.,  "n
 "dynamics",    "vor",            "vor",            "atmos_average",  "all", .true.,  "n
 "dynamics",    "tracer1",        "tracer1",        "atmos_average",  "all", .true.,  "n
 "dynamics",    "tracer2",        "tracer2",        "atmos_average",  "all", .true.,  "n
#"hs_forcing",  "teq",            "teq",            "atmos_average",  "all", .true.,  "n
```

5.3. The field table

The FMS [http://www.gfdl.noaa.gov/~fms] field [../src/shared/field_manager/field_manager.html] and tracer

managers [../src/shared/tracer_manager/tracer_manager.html] are used to manager tracers and specify tracer options. All tracers used by the model must be registered in an ASCII table called `field_table`. The field tables supplied with the dynamical cores are found in each `atm_dycores/exp/$dycore` [../exp] directory. There are no field tables provided for the barotropic or shallow-water models. In the runscript [#runscript], the user specifies the full path to the appropriate field table, and it is copied to file `field_table`.

The field table consists of entries in the following format. The first line of an entry should consist of three quoted strings. The first quoted string will tell the field manager [../src/shared/field_manager/field_manager.html] what type of field it is. The string "`tracer`" is used to declare a tracer field entry. The second quoted string will tell the field manager [../src/shared/field_manager/field_manager.html] which model the field is being applied to. The supported types at present are "`atmos_mod`" for the atmosphere model, "`ocean_mod`" for the ocean model, "`land_mod`" for the land model, and, "`ice_mod`" for the ice model. The third quoted string should be a unique tracer name that the model will recognize.

The second and following lines of each entry are called `methods` in this context. Methods can be developed within any module and these modules can query the field manager [../ src/ shared/ field_manager/ field_manager.html] to find any methods that are supplied in the field table. These lines can consist of two or three quoted strings. The first string will be an identifier that the querying module will ask for. The second string will be a name that the querying module can use to set up values for the module. The third string, if present, can supply parameters to the calling module that can be parsed and used to further modify values. An entry is ended with a backslash (/) as the final character in a row. Comments can be inserted in the field table by having a # as the first character in the line.

Here is an example of a field table entry for an idealized tracer called "`gunk`".

```
      "TRACER",     "atmos_mod",                "gunk"
      "longname",   "really bad stuff"
      "units",      "kg/kg"
      "advec_vert", "finite_volume_parabolic"
      "diff_horiz", "linear",                   "coeff=.30" /
```

In this example, we have a simple declaration of a tracer called "`gunk`". Methods that are being applied to this tracer include setting the long name of the tracer to be "`really bad stuff`", the units to "`kg/kg`", declaring the vertical advection method to be "`finite_volume_parabolic`", and the horizontal diffusion method to be "`linear`" with a coefficient of "0.30".

A `method` is a way to allow a component module to alter the parameters it needs for various tracers. In essence, this is a way to modify a default value. A namelist can supply default parameters for all tracers and a method, as supplied through the field table, will allow the user to modify the default parameters on an individual tracer basis.

The following web-based documentation describes the available `method_types` for the dynamical cores.

- B-grid: advection and filling [../src/atmos_bgrid/model/bgrid_advection.html#FIELD_TABLE]
- B-grid: horizontal damping [../src/atmos_bgrid/model/bgrid_horiz_diff.html#FIELD_TABLE]
- Spectral: all tracer options [../src/atmos_spectral/model/spectral_dynamics.html#FIELD_TABLE]

5.4. Namelist options

Many model options are configurable at runtime using namelist input. All FMS [http://www.gfdl.noaa.gov/~fms] modules read their namelist records from files called `namelists`. A module will read `namelists` sequentially until the first occurrence of its namelist record is found. Only the first namelist record found is used. Most (if not all) namelist variables have default values, so it is not necessary to list all namelist records in the `namelists` file. The runscripts provided set up the file `input.nml` by concatenating the core-specific namelist files.

- Summary of namelist records for the B-grid core:

bgrid_core_driver_nml [../src/atmos_bgrid/model/bgrid_core_driver.html#NAMELIST]	main namelist for the B-grid core. It controls the following options: time splitting, domain layout, polar filtering, pressure gradient, divergence damping, energy conservation, and restart file [#ic_restarts] format.
bgrid_advection_nml [../src/atmos_bgrid/model/bgrid_advection.html#NAMELIST]	advection scheme and tracer filling options
bgrid_horiz_diff_nml [../src/atmos_bgrid/model/bgrid_horiz_diff.html#NAMELIST]	horizontal damping order and coefficients
bgrid_sponge_nml [../src/atmos_bgrid/model/bgrid_sponge.html#NAMELIST]	top model level sponge coefficients
bgrid_integrals_nml [../src/atmos_bgrid/tools/bgrid_integrals.html#NAMELIST]	used to set the file name and output interval for global integrals
bgrid_cold_start_nml [../src/atmos_bgrid/tools/bgrid_cold_start.html#NAMELIST]	needed for cold-starting the model, only read when the restart file [#ic_restarts] does not exist

- Summary of namelist records for the spectral core:

spectral_dynamics_nml [../src/atmos_spectral/model/spectral_dynamics.html#NAMELIST]	main namelist for the spectral core
atmosphere_nml	atmospheric settings for the barotropic [../src/atmos_spectral_barotropic/atmosphere.html#NAMELIST] and shallow [../ src/ atmos_spectral_shallow/atmosphere.html#NAMELIST] configurations
barotropic_dynamics_nml [../src/atmos_spectral_barotropic/barotropic_dynamics.html#NAMELIST]	dynamical settings for the barotropic configuration
barotropic_physics_nml	physics settings for the barotropic configuration
shallow_dynamics_nml [../src/atmos_spectral_shallow/shallow_dynamics.html#NAMELIST]	dynamical settings for the shallow configuration

- Summary of namelist records for the finite-volume core:

fv_core_nml [../exp/fv/namelists]	main namelist for the finite-volume core

- Summary of namelist records for miscellaneous modules:

main_nml [../src/atmos_solo/atmos_model.html#NAMELIST]	sets time related variables such as model time step and duration of the run
fms_nml [../src/shared/fms/fms.html#NAMELIST]	sets FMS [http://www.gfdl.noaa.gov/~fms] infrastructure options
hs_forcing_nml [../src/atmos_param/hs_forcing/hs_forcing.html#NAMELIST]	sets parameters related to Held-Suarez forcing
shallow_physics_nml [../src/atmos_spectral_shallow/shallow_physics.html#NAMELIST]	setting for the shallow configuration of the bgrid or spectral models

- Examples of common namelist option changes:

Change the run length:
&main_nml days = 200 /

: The default run length set in the simple runscripts [#runscript] is relatively short. The run length is set in the namelist for the main program, which can be found in the runscripts [#runscript]. For example, the following change will increase the run length to 200 days.
:

Change the model resolution:
&bgrid_cold_start_nml
nlon=90, nlat=60, nlev=10 /
&spectral_dynamics_nml
lon_max=192, lat_max=96, num_fourier=63, num_spherical=64 /

: Because the simple test cases internally generate their initial state, the model resolution can be easily changed through namelist variables. For the B-grid core, modify the following namelist found in file atm_dycores/exp/bgrid/namelists [../exp/bgrid/namelists]. Note that nlon and

`nlat` must be even numbers, and too low a resolution may result in an unphysical solution. For a *T63* spectral core, modify the following namelist variables found in the file `atm_dycores/exp/spectral/namelists` [`../exp/spectral/namelists`].
:

Change the model time step:
&main_nml dt_atmos = 1200 /

: If the model resolution is changed it may be necessary to also change the model time step. The time step for the atmospheric model is set in the namelist for the main program, which can be found in the runscripts [#runscript]. For example, in `atm_dycores/exp/bgrid/fms_runscript` [`../exp/bgrid/fms_runscript`] the following will change the time step to 1200 seconds.
:

Change the domains stack size:
&fms_nml domains_stack_size = 600000 /

: If you increase the model resolution or are running on a small number of processors, you may get the error message "MPP_UPDATE_DOMAINS user stack overflow". In this case, increase the domain stack size found in the core-specific namelist files. The new stack size should be greater than or equal to the number printed in the error message.
:

Run the FV core on multiple processors:
&fv_core_nml layout = 1, $npes /

: Decomposition is not supported in the x direction, so the first value is 1, and $npes is the number of processors the code is run on.

- Finite-volume dynamical core namelist, fv_core_nml:
The FV core is set up so that the defaults generally produced the best results. For example, the model will automatically determine a most efficient time step size if n_split is absent from the namelist input or is set to ZERO. The tracer time step is dynamically determined every time step to maintain CFL condition in the north-south direction to be less than ONE. If the user is uncertain about any namelist input, please use the default.
1. Required namelist input from the user:
:

nlon (integer)

: east-west dimension (e.g., nlon=144 for M45 resolution). The resolution (degrees) is 360/nlon. Due to the use of FFT at high latitudes for polar filtering, nlon must be an even number.
:

mlat (integer)

: north-south dimension (mlat=90 for M45 resolution). The meridional resolution (degrees) is 180/(mlat-1). For better load balance, it is recommend that mlat be divisible by 3 (e.g., 60, 90, 180). However, any number would work.
:

nlev (integer)

: vertical dimension (total number of layers).
:

ncnst (integer)

: total number of tracers (including all physics tracers nt_phys). For the standard AM2 physics, ncnst=4.
2. Optional input:

2.1 Input that controls the advection operators:

The following "ORD" values determine the transport schemes to be used for different spatial directions and different prognostic variables. These options are mainly for testing/development purposes and a typical user should just use the defaults (Piecewise Parabolic Method with the monotonicity constraint as described in Lin 2004). Setting these values to 2 will force the model to use the 2nd order accurate Van Leer scheme instead.
:

iord_mt (integer)

: advection operator in the zonal direction for momentum.
:

iord_tm (integer)

: advection operator in the zonal direction for thermodynamics.
:

iord_tr (integer)

: advection operator in the zonal direction for tracers.
:

jord_mt (integer)

: advection operator in the meridional direction for momentum.
:

jord_tm (integer)

: advection operator in the meridional direction for thermodynamics.
:

jord_tr (integer)

: advection operator in the meridional direction for tracers.
:

kord_mt (integer)

: advection operator in the vertical direction for momentum.
:

kord_tm (integer)

: advection operator in the vertical direction for thermodynamics.
:

kord_tr (integer)

: advection operator in the vertical direction for tracers.

2.2 Miscellaneous optional input:
:

nt_phys (integer)

: total number of tracers needed by physics (4 in AM2).
:

pnats (integer)

: number of non-advected tracers. This is used by some chemistry packages in which not all tracers are advected.
:

n_split (integer)

: number of time splits for the Lagrangian (horizontal) dynamics. The model contains an algorithm for the automatic determination of the most efficient value. If instability arises, the user may want to increase the value of n_split (therefore, reducing the size of time step).
:

n_zonal (integer)

: loop decomposition in East-West direction. This may have some impact on the computational speed if the platform is cache based or is "OpenMP" capable.
:

map_dt (integer)

: remapping time step (equal to model time step). The option of using different remapping time step than the physics is currently not supported.
:

consv_te (real)

: energy fixer. Range[0,1.]. The default is ZERO (no energy conservation correction). Setting consv_te to ONE force the dynamics to maintain exact conservation of total energy for each time step.
:

restart_format (character)

: "native" (IEEE) or "netcdf".
:

adiabatic (logical)

: run the model without any physics (diabatic) forcing. This flag is mainly used to turn off the Held-Suarez in the "solo" mode.
:

full_phys (logical)

: run the model with full (AM2) physics with virtual effects (when computing geopotential).
:

fill (logical)

: activate a vertical filling algorithm for tracers (needed if physics/chemistry produced negatives).
:

adjust_dry_mass (logical)

: adjust the initial global mean dry mass to pre-set value.
:

n_spong (integer)

: total number of sponge layers (counting from top; default is 1). The horizontal transport scheme is set to the highly diffusive first order upwind scheme within the sponge layers.
:

n_diffu (integer)

: number of diffusive layers (counting from the bottom; default is 0). This mirrors the top sponge layers. This option may be useful only in certain idealized environment -- it is not recommended for general usage.
:

change_time (logical)

: ignore time stamp in the restart file. This option is useful for swapping restart files from a different time/ date.
:

a_div (real)

: dimensionless divergence damping parameters: D = a_div + b_div * cos(lat). The physical damping coefficient will be equal to D * dx * dy / dt.
:

b_div (real)

: dimensionless divergence damping parameters: D = a_div + b_div * cos(lat).
:

use_set_eta (logical)

: use set_eta(), instead of restart, for (ak,bk). This is useful for doing model model cold start. Several settings are available, ranging from 18 to 100 layers. The routine for setting the definition of the "eta" coordinate (for remapping) is located in src/atmos_fv_dynamics/tools/set_eta.f90 [../ src/ atmos_fv_dynamics/ tools/ set_eta.f90].
:

use_tendency (logical)

: use the tendency approach for updates. This option is no longer supported.
:

do_ch4_chem

: relax H2o to Haloe data between 1 and 10 mb. This is a rarely used option. It is mainly used for starting the model from a totally dry atmosphere.
:

pft_phys (logical)

: polar filter physical tendencies. This option may be useful for high resolution runs. Default is .F.
:

age_tracer (logical)

: transport the age tracer as the last tracer. This option is no longer supported.
:

print_freq (integer)

: print max/min of some selected fields (0: off; positive n: every n hours; negative n: every time step).
:

icase (integer)

: test case number if "SW_DYN" (shallow water dynamics) is defined during compilation. In this mode, nlev must be set to ONE.
:

layout (integer)

: computational layout (layout = 1, $npes). Decomposition is not supported in the x-direction and $npes is the number of processors.

5.5. Initial conditions and restart files

FMS [http://www.gfdl.noaa.gov/~fms] uses restart files to save the exact (bit-reproducible) state of the model at the end of a model run. The restart files are used as an initial condition to continue a model run from an earlier model integration. A module (or package of modules) will write data to a restart file if it is needed for a bit-reproducible restart. The input restart files (the initial conditions) are read from the `INPUT` subdirectory. The output restart files are written to the `RESTART` subdirectory. The simple runscripts [#runscript] create these two directories when setting up the model run.

The initial condition file specified in the simple runscripts [#runscript] is a cpio archive file that contains the restart files created by individual modules. The test cases provided with this release do not specify an initial condition file, but rather generate their initial states internally. The test case will however create output restart files, and a user may want to archive or move the output restart files so they can be used as an initial condition when continuing a model run.

To create a cpio archive file:

```
cd RESTART
/bin/ls *.res* | cpio -ov > OutputRestart.cpio
```

or, simply move the output files to the `input` directory:

```
rm INPUT/*.res*
mv RESTART/*.res* INPUT
mpirun -np 1 fms.exe # rerun the model
```

Because the restart file for the main program contains information about the current model time, there is no need to modify any namelist or input files before rerunning the model.

The restart file created by the B-grid core is called `bgrid_prog_var.res.nc` and the restart file created by the spectral core is called `spectral_dynamics.res.nc`. Here are some specific details about the restart file for each core.

Bgrid:

- May be written as (64-bit) netCDF [http:/ / www.unidata.ucar.edu/ packages/ netcdf/] files or (machine-dependent) native format files. This output option is set using a namelist variable.
- May be restarted from either netCDF [http://www.unidata.ucar.edu/packages/netcdf/] or native restart files. The model will look for both file types, no namelist variable is needed.
- The restart file contains the vertical grid, topography, and the prognostic variables.

- The model resolution is determined from the restart file.
- If no restart file is present, then a cold-start initialization is attempted. See the namelist options and on-line documentation for module `bgrid_cold_start` [../src/atmos_bgrid/tools/bgrid_cold_start.html].
- A separate restart file is written/read for atmospheric tracers. When the netCDF restart option is used a single netCDF [http://www.unidata.ucar.edu/packages/netcdf/] tracer restart file is written. When native-format restarts are written each tracer is saved to a separate file.

Spectral:

- Written only as (64-bit) netCDF [http://www.unidata.ucar.edu/packages/netcdf/].
- May be restarted from either netCDF [http://www.unidata.ucar.edu/packages/netcdf/] or native restart files. The model will look for both file types, no namelist variable is needed.
- The restart file contains the vertical grid, topography, and the prognostic variables.
- The model resolution is determined from namelist settings, resolution of restart data is checked against that set in namelist.
- Model is cold-started when no restart file is present. Additional namelists may optionally be supplied when cold-started. These allow the specification of the vertical coordinate, initial temperature of the atmosphere, and the surface height.
- A separate restart file is written/read for each atmospheric tracer [#fieldTable].

Finite Volume:

- May be written as (64-bit) netCDF [http:/ / www.unidata.ucar.edu/ packages/ netcdf/] files or (machine-dependent) native format files. This output option is set using a namelist variable.
- May be restarted from either netCDF [http://www.unidata.ucar.edu/packages/netcdf/] or native restart files. The model will look for both file types, no namelist variable is needed.
- The restart file contains the vertical grid, topography, and the prognostic variables.
- The model resolution is determined from namelist settings, resolution of restart data is checked against that set in namelist.
- Model is cold-started when no restart file is present.
- A separate restart file is written/read for each atmospheric tracer [#fieldTable].

5.6. mppnccombine

Running the FMS [http://www.gfdl.noaa.gov/~fms] source code in a parallel processing environment will produce one output netCDF [http://www.unidata.ucar.edu/packages/netcdf/] diagnostic file per processor. **mppnccombine** joins together an arbitrary number of data files containing chunks of a decomposed domain into a unified netCDF [http://www.unidata.ucar.edu/packages/netcdf/] file. If the user is running the source code on one processor, the domain is not decomposed and there is only one data file. **mppnccombine** will still copy the full contents of the data file, but this is inefficient and **mppnccombine** should not be used in this case. Executing **mppnccombine** is automated through the runscripts [#runscript]. The data files are netCDF [http:/ / www.unidata.ucar.edu/packages/netcdf/] format for now, but IEEE binary may be supported in the future.

mppnccombine requires decomposed dimensions in each file to have a `domain_decomposition` attribute. This attribute contains four integer values: starting value of the entire non-decomposed dimension range (usually 1), ending value of the entire non-decomposed dimension range, starting value of the current chunk's dimension range and ending value of the current chunk's dimension range. **mppnccombine** also requires that each file have a `NumFilesInSet` global attribute which contains a single integer value representing the total number of

chunks (i.e., files) to combine.

The syntax and arguments of **mppnccombine** are as follows:

```
mppnccombine [-v] [-a] [-r] output.nc [input ...]
        -v      print some progress information
        -a      append to an existing netCDF [http://www.unidata.ucar.edu/packages/netcd
        -r      remove the '.####' decomposed files after a successful run
```

An output file must be specified and it is assumed to be the first filename argument. If the output file already exists, then it will not be modified unless the option is chosen to append to it. If no input files are specified, their names will be based on the name of the output file plus the extensions '.0000', '.0001', etc. If input files are specified, they are assumed to be absolute filenames. A value of 0 is returned if execution is completed successfully and a value of 1 indicates otherwise.

The source of **mppnccombine** (mppnccombine.c) is packaged with the FMS [http://www.gfdl.noaa.gov/~fms] dynamical cores in the `atm_dycores/postprocessing` [../postprocessing] directory. **mppnccombine.c** is automatically compiled in the runscript when more than 1 processor is specified. It should be compiled on the platform where the user intends to run the FMS [http://www.gfdl.noaa.gov/~fms] Memphis atmospheric dynamical cores source code. A C compiler and netCDF [http://www.unidata.ucar.edu/packages/netcdf/] library are required for compiling **mppnccombine.c**.

6. Examining output

6.1. Model output

Output from a FMS [http://www.gfdl.noaa.gov/~fms] model run will be written to the directory where the model was run. FMS models write output in ASCII, binary, and netCDF [http://www.unidata.ucar.edu/packages/netcdf/] formats. ASCII or text output files have the `*.out` suffix. For example, files of the form `*integral.out` contain global integrals and `logfile.out` contains the namelist and revision number output. Note that the spectral model does not produce `*integral.out` files. Standard output and standard error messages created by the model may be directed to a file called `fms.out`. The diagnostics files, specified in the diagnostics table, are written as netCDF [http://www.unidata.ucar.edu/packages/netcdf/] files with the `*.nc` suffix. The output restart files are written to the subdirectory `RESTART` and will have the `*.res.nc` or `*.res` suffix.

You may download sample output data for comparison at https://fms.gfdl.noaa.gov/projects/fms/ under the "Files" tab. Each tar file expands to a directory containing a readme file along with netcdf and ascii output. The files `bgrid_output.tar.gz`, `fv_output.tar.gz` and `spectral_output.tar.gz` contain daily snapshots of surface pressure and time means of all fields over the 200 to 1200 day period. The file `bgrid_shallow_output.tar.gz` contains daily snapshots of surface pressure and time means of all fields over a 30 day period. The file `spectral_barotropic_output.tar.gz` contains 1000 days of diagnostic output with a 200 day spin-up period for the spectral barotropic model. `spectral_shallow_output.tar.gz` contains 30 days of diagnostic output for the spectral shallow water model.

6.2. Displaying the output

There are several graphical packages available to display the model output. These packages widely vary depending on factors, such as the number of dimensions, the amount and complexity of options available and the output data format. The data will first have to be put into a common format that all the package can read. FMS [http://www.gfdl.noaa.gov/~fms] requires the data to be stored in netCDF [http://www.unidata.ucar.edu/packages/netcdf/] format since it is so widely supported for scientific visualization. The graphical package is also depend-

ent upon the computing environment. This section will discuss a two-dimensional browser that is used on workstations, **ncview**. Please reference the GFDL Scientific Visualization Guide [http://www.gfdl.noaa.gov/products/vis/visguide.html] for information on additional graphical packages.

6.3. ncview

ncview is a visual browser for netCDF [http://www.unidata.ucar.edu/packages/netcdf/] data format files and displays a two-dimensional, color representation of single precision floating point data in a netCDF [http://www.unidata.ucar.edu/packages/netcdf/] file. You can animate the data in time by creating simple movies, flip or enlarge the picture, scan through various axes, change colormaps, etc. **ncview** is not a plotting program or an analysis package. Thus, contour lines, vectors, legend/labelbar, axis/tic-marks and geography are not included. The user is unable to perform X-Z or Y-Z cross-sections, unless there is a fourth dimension, time. Rather, **ncview's** purpose is to view data stored in netCDF [http://www.unidata.ucar.edu/packages/netcdf/] format files quickly, easily and simply.

ncview is capable of short user spin-up, fast cross sectioning, magnification, predefined color palettes which can be inverted and 'scrunched' to highlight high or low values, animation along the least quickly varying dimension only with speed control and printing. It also has the ability to read a series of files as input, such as a sequence of snapshot history files. A time series graph for variables pops up by clicking the mouse at a specific point. Other options include a mouse-selectable colormap endpoints with optional reset, map overlay and a filter for one-dimensional variables.

In **ncview**, the user can **<left-click>** on any point in a plot to get a graph of the variable verses time at that point. Also, **<Ctrl><left-click>** on any point to set the colormap minimum to the value at that point, while **<Ctrl><right-click>** on any point will set the colormap maximum to the value at that point. Use the "Range" button to set (or reset) the colormap min/max. For additional information on **ncview**, refer to the **ncview** UNIX manual page (**man ncview**) or the **ncview** homepage [http://meteora.ucsd.edu/~pierce/ncview_home_page.html].

7. Performance

The test cases provided with this release have been run on the SGI Altix Intel Itanium2 1.5 GHz (ifort.9.0-027/mpt-1.12-1) and Origin 3800 MIPS R14000 600 MHz (mipspro_743m/mpt_1900) large-scale clusters at the Geophysical Fluid Dynamics Lab [http://www.gfdl.noaa.gov]. The table below summarizes the performance for each of the test cases.

Model	Resolution	Run length (days)	# pe	Time (sec) on A
bgrid	N45 (144 x 90 x 20)	200	15	2303
bgrid_shallow	N45 (144 x 90)	200	15	130
fv	M45 (144 x 90 x 20)	200	15	2425
spectral	T42 (128 x 64 x 20)	200	16	384
spectral_barotropic	T85 (256 x 128)	200	16	63
spectral_shallow	T85 (256 x 128)	200	16	112